(19) Europäisches Patentamt
European Patent Office
Office européen des brev ts

(11) Publication number: **0 498 130 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **91403349.3**

(51) Int. Cl.5: **G06F 9/44**

(22) Date of filing: **10.12.91**

(30) Priority: **08.02.91 US 652820**

(43) Date of publication of application:
12.08.92 Bulletin 92/33

(84) Designated Contracting States:
DE FR GB

(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Furtney, David Allen**
1806 17 1/2 Street N. W.
Rochester, Minnesota 55901(US)
Inventor: **Ingebretson, Mark Edward**
2015 41st Street N. W., Apt. K35
Rochester, Minnesota 55901(US)
Inventor: **Simonson, Douglas Lee**
690 Chelsea Lane N. W.
Rochester, Minnesota 55901(US)

(74) Representative: **Obolensky, Michel et al**
c/o CABINET LAVOIX 2, place d'Estienne d'Orves
F-75441 Paris Cédex 09(FR)

(54) Apparatus and method for verifying compatibility of system components.

(57) Each component (101) of a plurality of interacting system components is associated with a version identifier (102). The version identifier (102) is stored in a location accessible by the other components. Each component independently reads the version identifier of every other component with which it must interact, and compares this value to an internally stored compatibility record (103) to determine whether the other component satisfies requirements for compatibility with the verifying component. Any component which detects an incompatibility signals an error to the system. In the preferred embodiment, the components are software modules, and the version identifier and compatibility record contain integer values. The compatibility record value represents the minimum level required of the module being verified for compatibility with the verifying module. Compatibility verification is accomplished by comparing the actual level of the module being verified with the minimum level in the compatibility record. If the actual level is equal to or greater than the minimum level, the module being verified satisfies compatibility requirements.

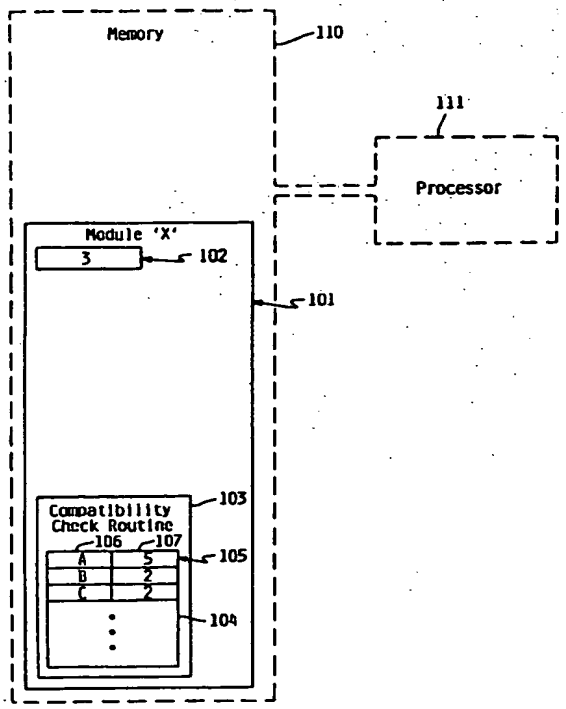

FIG. 1

EP 0 498 130 A2

The present invention relates to data processing component usage, and in particular to verifying that two or more interacting components of a digital data system are compatible with each other.

A modern computer system typically comprises a variety of different interconnected processors, each executing its own software. A single central processing unit (CPU) is typically the basic workhorse of the computer, but other processors control disk storage devices, printers, terminals, communications with other computers, etc. Even more remote and primitive processors may control other functions, such as monitoring sensors, input keypads, etc. In addition, multiple computer systems may be connected to a network, whereby they may communicate with each other.

In such a system, it is common for multiple system components to interact with each other. For exampl , interacting software modules may be running in separate processors or in a single processor. Processors which control peripheral functions, such as disk storage controllers, must be able to transmit data to and from the system CPU using som defined protocol. Therefore the software running in the peripheral processor must be compatible with the operating system software of the main system. Similarly, multiple software modules may be sharing the CPU and certain system resources. These modules may interact, for example, by sharing memory, and must be compatible in the way they acc ss and alter the shared data.

System components, and particularly software modules, are subject to frequent revision, either to correct known errors in the component or to enhanc its function. As a result of these frequent revisions, many versions of a component may exist, each with slightly different capabilities. Some versions of a component may therefore be incompatible with some versions of an interacting module. For example, if software modules A and B share a data structure, the addition of a new function to module A may require a new field in the data structure, and consequently require a new version of module B to support this new field.

Wh re multiple software modules are licensed from a single source and as a single package, the software developers can solve the problem of incompatible revision levels by guaranteeing that all modules shipped as part of the package are compatible with each other. A number of techniques exist in the art for tracking changes to software in the development environm nt and testing interacting modules before shipment to the customer to verify compatibility. However, where one module must interact with another from a different source, or that may hav be n acquired from the same source at a different time, the problem becomes more complex.

One approach to the problem is to allow the modules to execute and let standard system error recovery procedures detect incompatibility. While this may work in some cases, there is no guarantee that the system error recovery procedures will always detect incompatibility. For example, the incompatibility may be one which will corrupt data without generating an error condition. Another approach is to require that all modules be at the same level. While this guarantees module compatibility, it is unduly restrictive. Some modules may be compatible with each other even though not at the same level. This problem is particularly acute in the case of components which are not easily replaced not easily replaced, as for example, when software for a peripheral controller is stored in a read-only memory. There exists a need for a general method of verifying compatibility of system components, which will not prevent compatible components of differing version levels from interacting.

It is therefore an object of the present invention to provide an enhanced method and apparatus for verifying compatibility of a plurality of interacting system components.

It is therefore an object of the present invention to provide an enhanced method and apparatus for verifying compatibility of a plurality of interacting software modules.

Another object of this invention to provide a more reliable method and apparatus for detecting component incompatibility.

Another object of this invention is to reduce the instances of reported incompatibility among interacting components which are in fact compatible.

Another object of this invention is to reduce the need for replacing interacting components which are in fact compatible.

Another object of this invention is to reduce the cost of operating a computer system with multiple interacting system components.

Each component of a plurality of interacting system components is associated with a version identifier, which in the preferred embodiment is an integer value. The version identifier is stored in a location accessible by the other system components. Each component independently reads the version identifier of every other component with which it must interact, and compares this value to an internally stored compatibility record to determine whether the other component satisfies requirements for compatibility with the verifying component. Any component which detects an incompatibility signals an error to th system. It is possible under this arrangem nt for component A to satisfy compatibility criteria required by component B, but not vice v rsa.

In the preferred embodiment, the components

are software modules. It is assumed that a software module at any arbitrary level will satisfy all compatibility requirements that ar satisfied by each lower level of that same module. The compatibility record contains an integer representing the minimum level required of the module being verified for compatibility with the verifying module. Compatibility verification is accomplished by comparing the actual l vel of the module being verified with the minimum level in the compatibility record. If the actual l vel is equal to or greater than the minimum lev l, the module being verified satisfies compatibility requirements.

Fig. 1 shows a typical system component in its environment according to the preferred embodim nt of this invention;

Fig. 2 is a block diagram of the steps required in each component to verify compatibility with interacting components, according to the preferred embodiment;

Fig. 3 shows an example of a computer system using this invention.

Figure 1 shows a typical system component 101 in its environment according to the preferred embodiment of the present invention. In this preferred embodiment, the component is a software module 101 comprising a block of machine instructions and data stored in a memory 110. Module 101 xecutes in a programmable processor 111 coupl d to the memory. Module 101 interacts with one or more other software modules to perform some desired function. Module 101 contains version identifier 102, which in the preferred embodiment is an integer value. In the preferred embodiment, the version identifier is stored in a predefined memory location in the module, from which it can be read by other modules. Module 101 also contains compatibility check routine 103, which comprises component version table 104 and a series of instructions to access the version identifiers of int racting modules and compare these to values in table 104. Component version table contains one or more entries 105, each entry corresponding to a software module with which module 101 must interact. Each entry 105 comprises component class field 106 which identifies the interacting software module, and compatible version field 107 identifying the compatible versions of the interacting module. In the preferred embodiment, compatible version field 107 is an integer representing the minimum version level of the interacting module which satisfies requirements for compatibility with software module 101.

In the preferred embodiment, compatibility check routine 103 directly accesses the version identifier in each interacting module by accessing a memory location which is a fixed offset from the beginning of the interacting module. However, any number of alternative methods could be mployed. For example, in one alternate embodiment, a separat softwar module, callable by each of the interacting modules, could contain a fetch version function which accesses the location of the version identifiers. A software module would access the version identifier of another software module by issuing a call to the fetch version function, passing it the component name of the software module for which the version identifier is desired. In another alternative embodiment, an operating system reads all module version identifiers and creates a table of such identifiers in a location accessible to all modules. In another alternative embodiment, the module performing compatibility checking interrogates the interacting module for its version identifier via a pre-defined interface. The present invention only requires that each module in a set of interacting modules have means for accessing the version identifiers of the other modules in the set.

Figure 2 shows the steps required in each module of a set of interacting modules to verify compatibility of the set. In accordance with this invention, each module of the set must independently verify that each other module with which it interacts satisfies minimum requirements for compatibility with the module. Each module of the interacting set first accesses the version identifier of an interacting component (step 201), and accesses the minimum version level required for compatibility with the component from its component version table 104 (step 202). If the actual level represented by the version identifier is less than the minimum level from the table (step 203), the interacting component being verified is added to an error list (step 204). If there are more interacting components to check, the process repeats (step 205), until all interacting components have been checked. When all components have been checked, if any components are in the error list (not compatible with the component performing compatibility checking) (step 206), an error condition is signalled (step 207). The response of the system to such an error condition would be dependent on the application, but it would generally mean that the modules would not be permitted to execute, because results may be unpredictable.

In the preferred embodiment, it is assumed that a software module at any arbitrary level will include the capabilities and functions of each lower level of that same module that are required for compatibility with other modules. Thus, for purposes of compatibility, a higher level of a module being verified will always satisfy minimum requirements for compatibility with the verifying module if any lower level of th module being v rified satisfies such compatibility requirements. Accordingly, it is pref rred that the version identifier be an integer,

permanent read-only memory coupled to the processor, while the second is stored in an electrically erasable memory coupled to the processor. From time to time the computer's operating system will download a new version of the second module, but it is unable to alter the first. As a result of a new version of the operating system, it is possible that the first module will no longer satisfy requirements for compatibility with the second. In addition, it is possible that a pluggable device will be replaced or added without an upgrade to the operating system, and consequently that the second module will no longer satisfy requirements for compatibility with the first. In this simplified embodiment, it is not nec ssary to maintain a component version table with ntries for different modules, because only on other module need be checked. Therefore the minimum version level required to meet compatibility requirements is hard-coded into the software module as a constant value, eliminating the need for instructions that look up the value from a table. Each module compares the version level of the other module with this hard-coded constant to verify compatibility. If an incompatibility is detected, an appropriate message is sent to the operating system. The operating system may be able to cure the problem by downloading a different version of the second module; if not, appropriate action is tak n to inhibit interaction between the two software modules.

Although in the preferred embodiment, the system components being verified for compatibility are software modules, in an alternative embodiment this invention can be used to verify compatibility of hardware components or components that are combinations of hardware and software. For example, el ctronic circuit cards in a computer system ar also subject to frequent revision for the same reasons that software modules are. In accordance with this alternative embodiment, a circuit card comprises a programmable processor executing a software module. A permanent version identifier is associated with the combination of the card and its software. The card verifies compatibility with interacting circuit cards as described above for software modules.

Although a specific embodiment of the invention has been disclosed along with certain alternatives, it will be recognized by those skilled in the art that additional variations in form and detail may be made within the scope of the following claims.

Claims

1. A method for v rifying that a plurality of system components xecuting on a computer syst m are compatible with each other, wherein a component version is associated with ach of said plurality of components, and wherein each of said components is associated with a component version identifier identifying the component version of said component, said method being characterized in that it comprises the steps of:
   - obtaining the component version identifier (102) associated with a first (101) of said plurality of components;
   - determining whether the component version identified by said component version identifier (102) associated with said first module satisfies requirements for compatibility with a second of said plurality of components;
   - obtaining the component version identifier associated with said second of said plurality of components; and
   - determining whether the component version identified by said component version identifier associated with said second component satisfies requirements for compatibility with said first (101) of said plurality of components.

2. The method for verifying compatibility according to claim 1, characterized in that each component version identifier (102) associated with a component (101) is stored in a fixed externally accessible location within the respective component (101).

3. The method for verifying compatibility according to claim 1, characterized in that each said component (101) is a software module.

4. The method for verifying compatibility according to claim 3, characterized in that said step of determining whether the component version identified by said component version identifier (102) associated with said first component (101) satisfies requirements for compatibility with a second component comprises the steps of: (a) accessing compatible component version identifier information in said second component, and (b) comparing said component version identifier (102) associated with said first component (101) with said compatible component version identifier information in said second component; and said step of determining whether the component version identified by said component version identifier associated with said s cond component satisfies requirements for compatibility with said first component (101) comprises the steps of: (a) accessing compatible component version identifi r information in said first component (101), and (b) comparing said component version

identifier associated with said second component with said compatible component version identifier information in said first component (101).

5. The method for verifying compatibility according to claim 4, characterized in that each component version identifier associated with a component is stored in a fixed externally accessible location within the respective component.

6. The method for verifying compatibility of claim 4, characterized in that said compatible compon nt version identifier information in said first and second components comprises in each of said components a table (104) of compatible component version identifier values.

7. The method for verifying compatibility of claim 4, characterized in that said component version identifier associated with each of said first and second components comprises an ordered value representing a component version level of the respective component;
said compatible component version identifier information contained in each of said first and second components comprises an ordered value representing a minimum compatible component version level; said step of comparing said component version identifier associated with said first component (101) with said compatible version identifier information contained in said second component comprises comparing said ordered value representing the component version level of said first component (101) with said ordered value representing a minimum compatible component version level in said second component and determining that said first component satisfies requirements for compatibility with said second component if said ordered value representing the component version level of said first component is greater than or equal to said ordered value representing a minimum compatible component version level in said second component; and said step of comparing said component version identifier associated with said second component with said compatible version identifier (102) information contained in said first component (101) comprises comparing said ordered value representing the compon nt version level of said s cond component with said ordered value representing a minimum compatible component v rsion level in said first component and determining that said second component satisfies requir ments for compatibility with said first component if said ordered value representing the component version level of said second component is greater than or equal to said ordered value representing a minimum compatible component version level in said first component.

8. A first system component of a plurality of interacting components of a computer system, wherein a component version is associated with each of said plurality of interacting components, said first system component being characterized by :
   - identifier fetching means for obtaining the component version identifier associated with a second component of said plurality of interacting components;
   - means for accessing compatible component version identifier information for said first component (101);
   - comparing means for comparing said component version identifier obtained by said identifier fetching means with said compatible component version identifier information to determine whether the component version identified by said component version identifier satisfies requirements for compatibility with said first component (101).

9. The first system component according to claim 8, characterized in that said component version identifier (102) associated with said first component (101) is stored in a fixed location within said first component accessible to means for accessing said component version identifier contained in said second component.

10. The first system component according to claim 8, characterized in that said component is a software module.

11. The first system component according to claim 8, characterized in that each said component version identifier associated with a component comprises an ordered value representing a component version level of the respective module; said compatible component version identifier information associated with said first component (101) comprises an ordered value representing a minimum compatible component version level; and said comparing means compares said ordered value representing the component version level with said ordered value repr senting a minimum compatible component v rsion l vel in said first component (101) and determines that said component version satisfies requirements for compatibility with said first component if said ordered valu

representing the component version level is greater than or equal to said ordered value representing a minimum compatible component version level in said first component.

12. The first system component according to claim 11, characterized in that said component version identifier (102) associated with said first component (101) is stored in a fixed location within said first component accessible to means for accessing said component version identifier
contained in said second component.

13. The first system component according to claim 11, characterized in that said component is a software module.

14. A computer system comprising:
- at least one programmable processor (111);
- a plurality of interacting system components (101),
wherein a respective component version identifier (102) is associated with each of said plurality of interacting components;
- means for obtaining the component version identifier (102) associated with a first (101) of said plurality of components;
- means for determining whether the component version identified by said component version identifier (102) associated with said first module satisfies requirements for compatibility with a second of said plurality of components;
- means for obtaining the component version identifier associated with said second of said plurality of components; and
- means for determining whether the component version identified by said component version identifier associated with said second component satisfies requirements for compatibility with said first of said plurality of components.

15. The computer system according to claim 14, characterized in that each component version identifier associated with a component is stored in a fixed externally accessible location within the respective component.

16. The computer system according to claim 14, characterized in that each said interacting system component is a software modul .

17. The computer system according to claim 16, characterized in that : said means for determining whether the component version identified by said component version identifier associated with said first component (101) satisfies requir ments for compatibility with a second component comprises: (a) means for accessing compatible component version identifier information in said second component, and (b) means for comparing said component version identifier associated with said first component (101) with said compatible component version identifier information in said second component; and said means for determining whether the component version identified by said component version identifier associated with said second component satisfies requirements for compatibility with said first component comprises: (a) means for accessing compatible component version identifier information in said first component, and (b) means for comparing said component version identifier associated with said second component with said compatible component version identifier information in said first component.

18. The computer system according to claim 17, characterized in that each component version identifier associated with a component is stored in a fixed externally accessible location within the respective component.

19. The computer system according to claim 17, characterized in that said component version identifier associated with each of said first and second components comprises an ordered value representing a component version level of the respective component; said compatible component version identifier information contained in each of said first and second components comprises an ordered value representing a minimum compatible component version level; said means for comparing said component version identifier associated with said first component with said compatible version identifier information contained in said second component compares said ordered value representing the component version level of said first component (101) with said ordered value representing a minimum compatible component version level in said second component and determines that said first component (101) satisfies requirements for compatibility with said second component if said ordered value representing the component version level of said first component is greater than or equal to said ordered value representing a minimum compatible component version level in said second component; and said means for comparing said component v rsion id ntifi r associated with said second component with said

compatible version identifier (102) information contained in said first component compares said ordered value representing the component version level of said second component with said ordered value representing a minimum compatible component version level in said first component and determines that said second component satisfies requirements for compatibility with said first component (101) if said ordered value representing the component version level of said second component is greater than or equal to said ordered value representing a minimum compatible component version level in said first component.

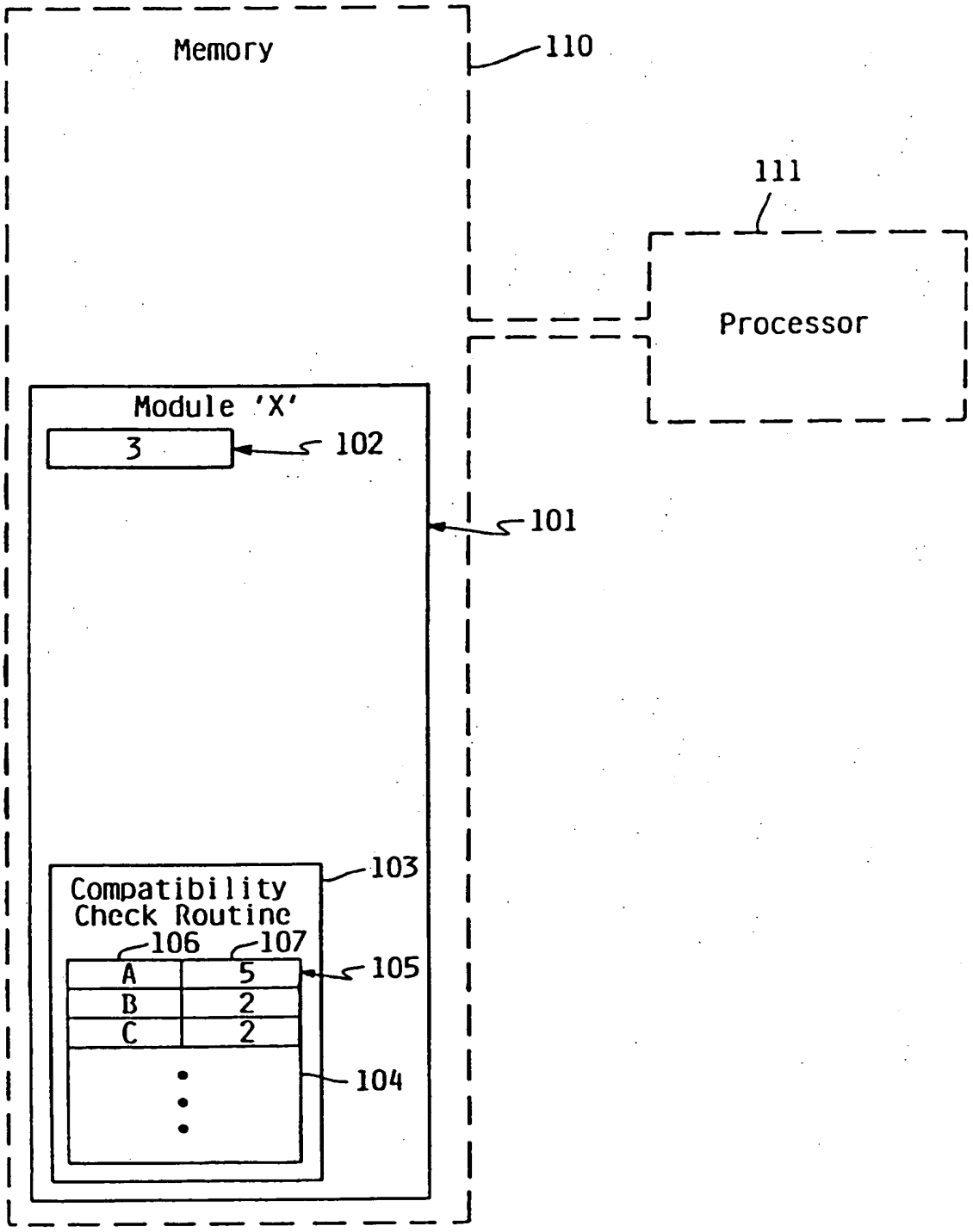
Memory
110
111
Processor
Module 'X'
3
102
101
Compatibility
Check Routine
103
106
107
A
5
B
2
C
2
105
104

FIG. 1

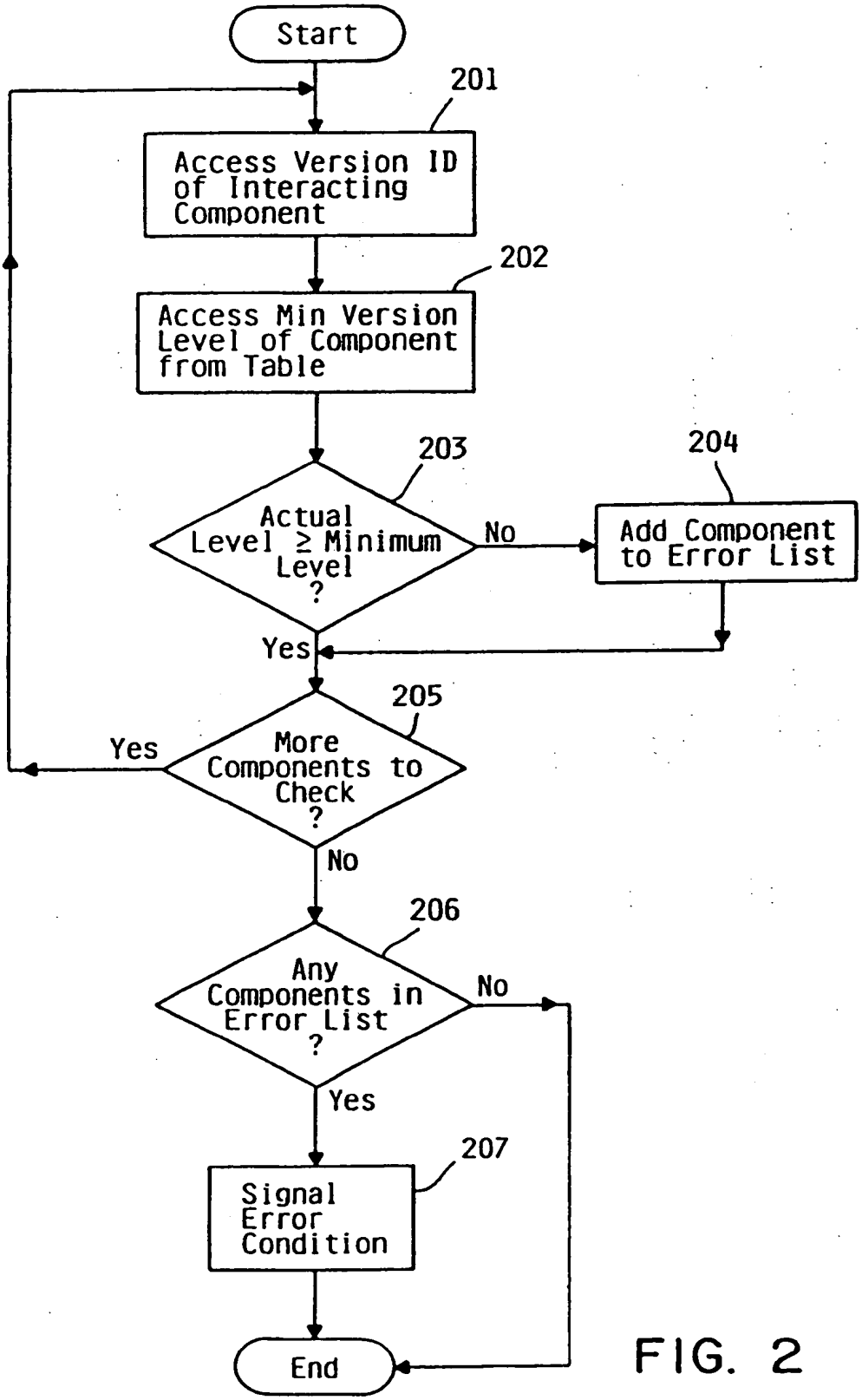
Start
201
Access Version ID of Interacting Component
202
Access Min Version Level of Component from Table
203
Actual Level ≥ Minimum Level ?
No
204
Add Component to Error List
Yes
205
More Components to Check ?
Yes
No
206
Any Components in Error List ?
No
Yes
207
Signal Error Condition
End

FIG. 2

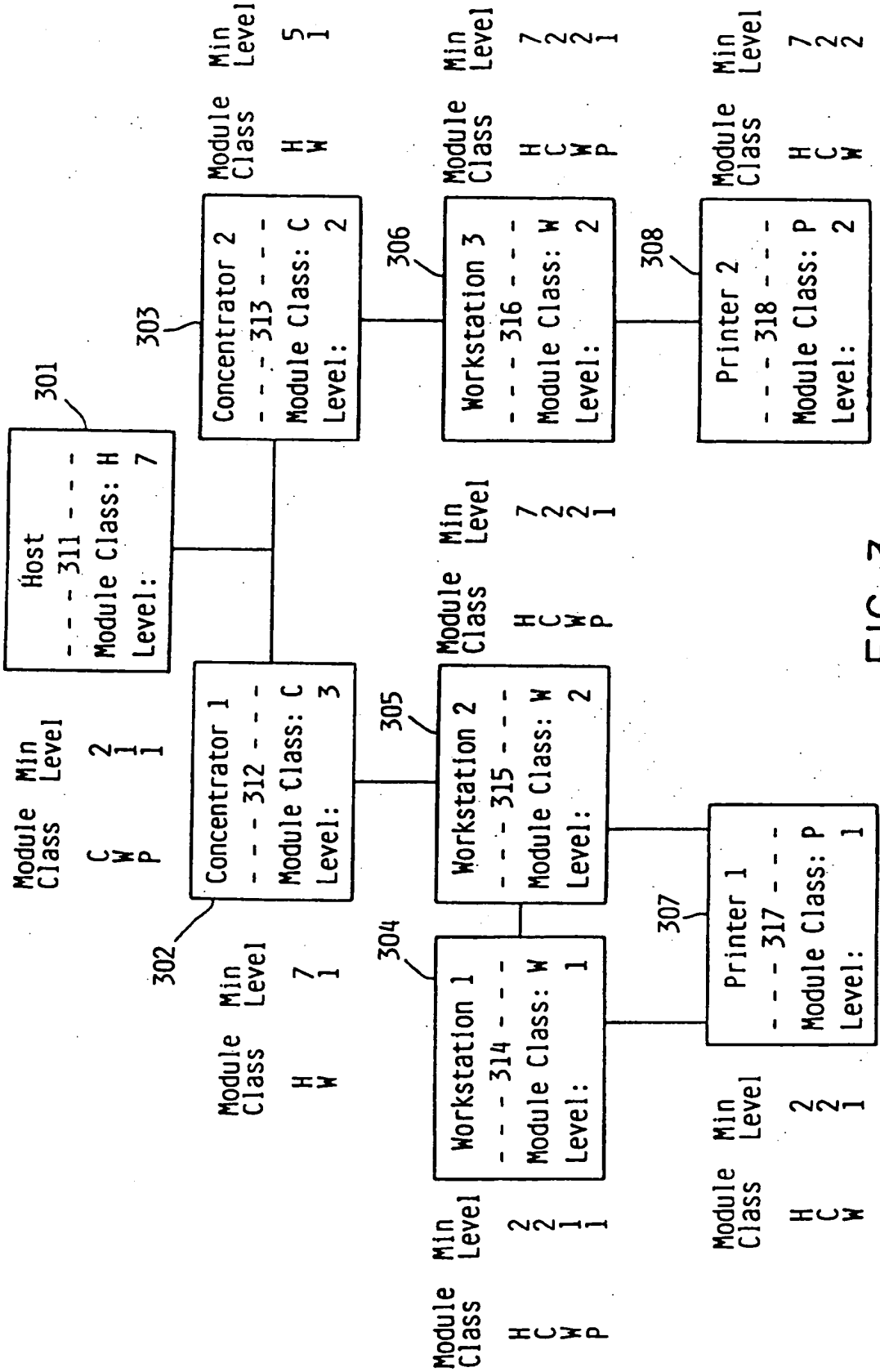
Host
- - - 311 - - -
Module Class: H
Level: 7
301
Module Class Min Level
C 2
W 1
P 1
Concentrator 1
- - - 312 - - -
Module Class: C
Level: 3
302
Module Class Min Level
H 7
W 1
Concentrator 2
- - - 313 - - -
Module Class: C
Level: 2
303
Module Class Min Level
H 5
W 1
Workstation 1
- - - 314 - - -
Module Class: W
Level: 1
304
Module Class Min Level
H 2
C 2
W 1
P 1
Workstation 2
- - - 315 - - -
Module Class: W
Level: 2
305
Module Class Min Level
H 7
C 2
W 2
P 1
Workstation 3
- - - 316 - - -
Module Class: W
Level: 2
306
Module Class Min Level
H 7
C 2
W 2
P 1
Printer 1
- - - 317 - - -
Module Class: P
Level: 1
307
Module Class Min Level
H 2
C 2
W 1
Printer 2
- - - 318 - - -
Module Class: P
Level: 2
308
Module Class Min Level
H 7
C 2
W 2

FIG. 3

(19) Europäisches Patentamt
European Patent Office
Offic européen des brevets

(11) Publication number: **0 498 130 A3**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **91403349.3**

(51) Int. Cl.5: **G06F 9/44**

(22) Date of filing: **10.12.91**

(30) Priority: **08.02.91 US 652820**

(43) Date of publication of application:
12.08.92 Bulletin 92/33

(84) Designated Contracting States:
DE FR GB

(88) Date of deferred publication of the search report:
03.02.93 Bulletin 93/05

(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Furtney, David Allen**
1806 17 1/2 Street N. W.
Rochester, Minnesota 55901(US)
Inventor: **Ingebretson, Mark Edward**
2015 41st Street N. W., Apt. K35
Rochester, Minnesota 55901(US)
Inventor: **Simonson, Douglas Lee**
690 Chelsea Lane N. W.
Rochester, Minnesota 55901(US)

(74) Representative: **Obolensky, Michel et al**
c/o CABINET LAVOIX 2, place d'Estienne d'Orves
F-75441 Paris Cédex 09(FR)

(54) Apparatus and method for verifying compatibility of system components.

(57) Each component (101) of a plurality of interacting system components is associated with a version identifier (102). The version identifier (102) is stored in a location accessible by the other components. Each component independently reads the version identifier of every other component with which it must interact, and compares this value to an internally stored compatibility record (103) to determine whether the other component satisfies requirements for compatibility with the verifying component. Any component which detects an incompatibility signals an error to the system. In the preferred embodiment, the components are software modules, and the version identifier and compatibility record contain integer values. The compatibility record value represents the minimum level required of the module being verified for compatibility with the verifying module. Compatibility verification is accomplished by comparing the actual level of the module being verified with the minimum level in the compatibility record. If the actual level is qual to or greater than the minimum level, the module being verified satisfies compatibility requirements.

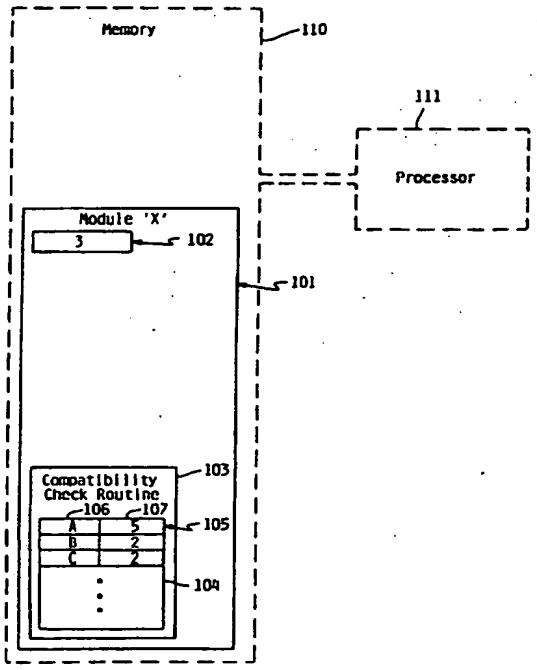

FIG. 1

EP 0 498 130 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 91 40 3349

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
X	EP-A-0 275 448 (SIEMENS AKTIENGESELLSCHAFT BERLIN UND MÜNCHEN) * the whole document *	1-19	G06F9/44
X	EP-A-0 217 351 (KABUSHIKI KAISHA TOSHIBA) * column 2, line 18 - line 36 * * column 3, line 12 - column 45; figure 1 *	1-19	
A	EP-A-0 282 149 (NORTHERN TELECOM LIMITED) * page 2, line 20 - line 45 * * page 3, line 28 - page 4, line 13 *	1-19	
A	IBM TECHNICAL DISCLOSURE BULLETIN vol. 33, no. 1B, June 1990, ARMONK, NY, US pages 343 - 345 ANON. 'RELEASE N -> N-1 VERSION NUMBER METHOD WITH DOWNLEVEL CODE AND FIX DIRECTION' * the whole document *	1-19	
A	EP-A-0 038 147 (THE POST OFFICE)	1-19	TECHNICAL FIELDS SEARCHED (Int. Cl.5)
A	CONFERENCE ON SOTWARE MAINTENANCE - 1988 - PROCEEDINGS 27 October 1988, PHOENIX, AZ, US pages 126 - 131 L. H. GREENE 'SELF-IDENTIFYING SOFTWARE' * page 127, right column, line 50 - page 128, left column, line 25 *	2,5,9, 12,15	G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	30 NOVEMBER 1992	R. ABRAM

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0403)